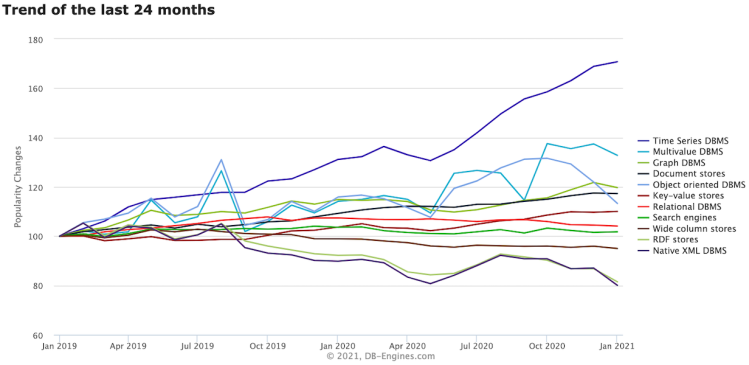
# Why We Need a Time-series Database

## Background

Currently, the storage and processing of temporal big data are often handled by relational databases, but the inherent disadvantages of relational databases make it impossible to store and query data efficiently. By using a special storage method, the temporal big data solution enables efficient storage and fast processing of massive temporal big data, which is an important technology for solving massive data processing. The technology uses a special data storage method, which greatly improves the processing capability of time-related data, halves the storage space and greatly increases the query speed compared to relational databases. The superior query performance of time series functions far exceeds that of relational databases. So time series databases are becoming more and more popular.



## Definition

Time series database is mainly used to refer to the processing of time-tagged (sequential change in time, i.e. time serialization) data, with the time tag data is also called time series data

## Characteristics of time-series data

* Time stamped data
* Structured data
* Data source is like a stream
* Data rate is pretty stable
* Immutable
* More write than read operations
* Data is rarely deleted or updated
* There is always retention policy
* Real-time data computing is desired
* Query is always in time and space range

## Characteristics

Specially optimized for processing time series data

This type of data is sorted in time

Relational databases often have difficulty handling this type of data due to its large size (hence the importance of sharding and scaling) or complex logic (lots of aggregation, fetching up, drilling down)

Time series data is divided into two categories by characteristics

High frequency low retention (data collection, real-time presentation)

Low frequency high retention period (data presentation, analysis)

By frequency

Regular interval (data collection)

Irregular interval (event-driven)

A few premises for time series data

Individual data is not important

Data is hardly ever updated, or deleted (only when expired data is deleted), and new data is the most recent data by time

The same data is considered to be the same data if it appears multiple times

## Overview

Time series datasets are relatively large and uniform compared to other datasets―usually being composed of a timestamp and associated data. Time series datasets can also have fewer relationships between data entries in different tables and don't require indefinite storage of entries. The unique properties of time series datasets mean that time series databases can provide significant improvements in storage space and performance over general purpose databases. For instance, due to the uniformity of time series data, specialized compression algorithms can provide improvements over regular compression algorithms designed to work on less uniform data. Time series databases can also be configured to regularly delete old data, unlike regular databases which are designed to store data indefinitely. Special database indices can also provide boosts in query performance.

## What makes a TSDB popular?

First, datasets are large and getting larger. Log files are measured in petabytes now, and they’re growing. Devices from the so-called internet of things (IoT) are proliferating, and they’re often designed to rely on a central service for analysis and presentation. Sense.com, for instance, collects information on electrical consumption in houses millions of times per second. When these bits are reported, Sense.com’s central database must store enough data to be useful but not enough to overwhelm the storage.

The time-series datasets often have fewer relationships between data entries in different tables that require transaction-based locking to avoid inconsistencies. Most of the data packets contain a timestamp, several sensor readings, and not much more.

This allows special indices to speed queries like the number of events in a day, week, or other time period. Good time-series indices can offer quick answers to statistical questions about ranges of data.

The databases can also offer some support because many of the maintenance chores are regular and easy to automate. The databases can automatically dispose of old data while delivering only fresh statistics. While standard databases are designed to store data forever, time-series databases can be configured to give data elements a specific time to live. Others will use a round-robin algorithm to store a fixed set.

As time goes by, the databases deploy specialized compression functions that will store time-series data in less space. If sensor readings don’t change from millisecond to millisecond, there’s no reason to store another copy of the same value. Timescale.com, for instance, boasts of 94%-97% saving in storage thanks to compression algorithms tuned to the regular data patterns.

## **The problem of traditional relational database to store** time-series data

Once you have the time-series data, where should you store it? Let's first look at what problems traditional relational database solutions encounter when storing temporal data.

Many people may think that adding a timestamp column to a traditional relational database can be used as a temporal database. It is true that it is fine when the data volume is small. However, temporal data is often generated by millions or even tens of millions of end devices, and the writing concurrency is relatively high, which is a massive data scenario.

MySQL has the following problems in the massive temporal data scenario.

High storage cost: poor compression for temporal data, which requires a large amount of machine resources.

High maintenance cost: single-machine system, which requires manual sub-banking and sub-tabling in the upper layer, and high maintenance cost.

Low write throughput: single machine write throughput is low, and it is difficult to meet the write pressure of ten million temporal data.

Poor query performance: suitable for transaction processing and poor performance for aggregation and analysis of massive data.

In addition, using Hadoop ecology (Hadoop, Spark, etc.) to store time-series data will have the following problems.

High data latency: offline batch systems, where data takes hours, or even days, from generation to analyzability.

Poor query performance: not good use of indexes, reliance on MapReduce tasks, query time consuming generally at the minute level.

It can be seen that the following problems need to be solved for a time-series database.

Writing of time-series data: how to support the writing of tens of millions of data points per second.

Time-series data reading: How to support grouping and aggregation of hundreds of millions of data at the second level.

Cost-sensitive: The problem of cost is brought by the massive data storage. How to store these data at lower cost will become the top priority of the temporal database.

## Advantages of Time-Series Database

Time-series database products are invented to solve the shortcomings and deficiencies of traditional relational databases in the storage and analysis of time-series data, and these products are uniformly categorized as time-series databases. \*\*\* Optimization of write, storage and query processes are carried out for the characteristics of time-series data, and these optimizations are closely related to the characteristics of time-series data.

Storage cost.

Using the characteristics of time increment, dimensional repetition and smooth change of indicators to reasonably select the coding compression algorithm and improve the data compression ratio.

Save storage space by pre-decreasing precision and doing aggregation on historical data.

High concurrent writes.

Writing data in batches to reduce network overhead.

Data is first written to memory and then periodically dumped to immutable file storage.

Low query latency, high query concurrency.

Optimizing common query patterns and reducing query latency through techniques such as indexing.

Improve query concurrency through caching, routing and other techniques

Why is Time-Series Database needed?

Data volume is huge (in most cases, it is generated automatically)

* Data ingestion rate becomes a challenge
* Query latency becomes a challenge
* Storage becomes a challenge

Special data analysis is required

* Down sampling
* Interpolation
* Time weighted average

…………………

## The Challenge of Time-Series Databases

The biggest challenge of the temporal database is how to accomplish various tasks in the massive data, which can be divided into the following aspects

Time-series data read-in: Because sometimes the frequency of data writing is very high, reaching tens of thousands of times per second, how to support high-intensity writing is the challenge of the time-series database.

Time-series data reading: Because of the high-frequency nature of time-series data, how to support high-intensity grouping and aggregation operations is also a major challenge.

Storage cost problem: Because of the huge amount of data, you need to consider how to store data at low cost.

Database ease of use requirements: how to design a temporal database so that the current it staff can quickly get started.

## Is there anything a TSDB can’t do?

In a sense, all databases are time-series databases because they maintain a log of the transactions that build up the table. The real question is which applications need to track how data changes over time. Many traditional databases were concerned only with the current state. They tracked, for instance, how many empty seats were left on the airplane. That’s the most important detail for selling tickets.

But sometimes there are hidden opportunities in even these applications. For instance, tracking when the tickets are sold can help pricing strategies in the future because airlines can know whether demand is running ahead or behind historical norms. In this sense, even traditional applications that don’t seem to need to track changes over time might be improved. The time-series databases might just be an opportunity.